AG III
SB

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application No.	10/004,196
Filing Date	November 14, 2001
First Named Inventor	Jose Fernandez
Art Unit	2123
Examiner Name	Guill, Russell L.
Total Number of Pages in This Submission	24
Attorney Docket Number	42390P10728

ENCLOSURES *(check all that apply)*

- ☐ Fee Transmittal Form
  - ☐ Fee Attached
- ☐ Amendment / Reply
  - ☐ After Final
  - ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement
  - ☐ PTO/SB/08
- ☐ Certified Copy of Priority Document(s)
- ☐ Response to Missing Parts/ Incomplete Application
  - ☐ Basic Filing Fee
  - ☐ Declaration/POA
- ☐ Response to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s)
  - ☐ Landscape Table on CD
- ☐ After Allowance Communication to TC
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☒ Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- ☐ Proprietary Information
- ☐ Status Letter
- ☒ Other Enclosure(s) *(please identify below):*

Return postcard

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm *or* Individual name	Gordon R. Lindeen III, Reg. No. 33,192 BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
Signature	
Date	December 4, 2007

CERTIFICATE OF MAILING/TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Typed or printed name	**Debbie Casias**		
Signature	Debbie Casias	Date	**December 4, 2007**

Our Docket No.: 4210728

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	
Jose Fernandez	Examiner: Guill, Russel L.
Application No.: 10/004,196	Art Group: 2123
Filed: November 14, 2001	
For: Generic Persistence Engine and Related Methods	

Mail Stop: Appeal Brief - Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REPLACEMENT APPEAL BRIEF
IN SUPPORT OF APPELLANT'S APPEAL
TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

Sir:

Applicant (hereinafter "Appellant") hereby submits this Replacement Appeal Brief (hereinafter "Brief") in support of its appeal from a final decision by the Examiner, mailed May 11, 2007, in the above-referenced Application. The replacement is due to the Notification of Non-Compliant Appeal Brief mailed 11/05/2007. Appellant respectfully requests consideration of this appeal by the Board of Patent Appeals and Interferences (hereinafter "Board") for allowance of the above-captioned patent application.

An oral hearing is not desired.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The invention is assigned to Intel Corporation of 2200 Mission College Boulevard, Santa Clara, California 95052.

II. RELATED APPEALS AND INTERFERENCES

To the best of Appellant's knowledge, there are no appeals or interferences related to the present appeal that will directly affect, be directly affected by, or have a bearing on the Board's decision.

III. STATUS OF CLAIMS

Claims 1-5 and 7-49 are currently pending in the above-referenced application. No claims have been allowed. All pending claims were rejected in the Final Office Action mailed May 11, 2007, and are the subject of this appeal.

All pending claims stand rejected under 35 U.S.C. § 103.

IV. STATUS OF AMENDMENTS

In response to the Final Office Action mailed on May 11, 2007, rejecting claims 1-5 and 7-49, Appellant timely filed a Notice of Appeal on August 10, 2007.

A copy of all claims on appeal is attached hereto as Appendix A.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The following paragraphs of the originally filed specification are believed to be instructive in considering the present application.

The first paragraph of the Detailed Description recites as follows

Described herein is a generic persistence engine (GPE) method and apparatus that allows persistent data sources, including software components having persistent data in diverse formats and data model structures, to participate in the persistent data management scheme of a running system without placing the system in an inactive or unavailable state

The first paragraph of the Background recites as follows:

Persistence is a property related to programming languages by which created objects continue to exist and variables continue to retain their values between runs of a program. Persistence allows data to have lifetimes that vary from transient to indefinite. Thus, persistence is particularly important in internet-programming languages, object-oriented databases, and data warehousing. Persistent programming languages offer an alternative to applications that require more than traditional database support.

Claim 1 refers to a method with the following elements:

"receiving a persistence package at a running system from one of a plurality of different software components, the persistence package including persistent data and metadata, the software components having persistent data in different formats that are foreign to the running system; See page 11, line 21, item 702.

"extracting persistent data and metadata from the persistence package, the persistent data relating to diverse types of objects constructed at runtime of the software component and needed during more than one invocation of the software component, the metadata describing the persistent data, and comprising, at least in part, a description of the format of the persistent data ; See page 7, line 24, item 414.

"establishing, based on the extracted metadata, a transform for a storage format and a storage location for the persistent data ; See page 11, line 24.

"applying the transform to the persistent data to format the persistent data without using the software component from which the persistence package was received from the format of the software component into a storage format that is compatible with the receiving system and with a storage device of the running system independent of the software component; See page 8, line 4, item 424.

"storing the persistent data in the storage device in the storage format ." See page 11, line 26

Claim 9 refers to an apparatus with a data storage device, see page 8, line 11, and a running receiving system 406, see page 7, line14.

Claim 29 refers to an apparatus with a communications interface, see page 7, line13, a data model description receiver, see page 7, line 26, a set of transforms, see page 7, line 30, a data model comparator, see page 7, line 29, a transform generator, see page 7, line 31, a storage device, see page 8, line 11, a transform engine, see page 7, line 29, and storing interface, see page 8, line 9.

Claim 36 is a method claim having elements that are described in the same sections of the specification as set forth for Claim 1. It has the following elements:

"receiving a data model description at a running system from one of a plurality of different software components that are foreign to the running system, the software components having persistent data in accordance with different data models, the persistent data relating to diverse types of objects, the data model description describing the persistent data, and comprising, at least in part, a description of the format of the persistent data; See page 11, line 21, item 702

"comparing the data model description to a preexisting data model independent of the software component from which the data model description is received; See page 7, line 7, item 318, page 9, line 1, item 504, page 11, lines 17-19

"assembling a transform at the running system independent of the software component from which the data model description is received based on the data model description and the comparison to establish a storage format for persistent data during runtime of a system; See page 11, line 24.

"applying a transform to format persistent data for storage from the format of the software component into a storage format that is compatible with a storage device independent of the software component; and See page 8, line 4, item 424

"storing the formatted persistent data at the storage device." See page 11, line 26

Claim 38 is a method claim that is dependent on Claim 36 and adds the element of "assembling a transform includes approximating a preexisting data model." see page 12, line 5, item 814.

Claim 43 is a *Beauregard* claim having elements that are described in the same sections of the specification as set forth for Claim 1, above. A machine-readable medium is shown for example as items 104, 106, page 5, lines 22-24, item 306, page 6, line 30, etc. The operations of the machine are as follows:

"receive persistent data having a model structure from one of a plurality of different software components that are foreign to the machine and the machine-readable medium, the software components having persistent data in different model structures, the persistent data relating to diverse types of objects; See page 11, line 21, item 702.

"receive metadata comprising at least in part a description of the model structure, the metadata describing the persistent data and comprising, at least in part, a description of the format of the persistent data ; See page 7, line 24, item 414. and

"establish, using the metadata and without using the software component from which the persistence package was received, a storage format and a storage location for the persistent data; See page 11, line 28, item 800 and

"apply the established storage format to the persistent data to format the persistent data for storage from the format of the software component into a storage format that is compatible with the machine and with a storage device independent of the software component." See page 8, line 4, item 424.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Claims 29-35 stand rejected under 35 U.S.C. §101 as directed to non-statutory subject matter

B. Claims 1-3, 7-13, 15-20 and 22-23 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Manning, U.S. Patent Publication No. 2002/0103829, ("Manning"), in view of Morgenstern, U.S. Patent No. 5,970,490 ("Morgenstern").

The remaining rejections rely on this rejection. Only this first rejection is to be reviewed.

VII. ARGUMENT

A. 35 U.S.C. §101 Rejection

The Examiner has rejected claims 29-35 under 35 U.S.C. §101 as directed to non-statutory subject matter. Claim 29, for example, recites a communications interface, a storage device, and a storing interface. These are all conventionally implemented as hardware. Other elements, such as the data model comparator, the transform generator and the transform engine can also be implemented as hardware.

The Examiner would appear to be requiring a processor. Such a requirement is not based on statute, rule, regulation or precedent. The comparator, generator and engine all provide sufficient structure to meet the statutory subject matter requirements.

Applicant notes for the record that this rejection should have been presented in the first Office action but was only presented much later and that the MPEP cautions against piecemeal examination of this kind.

B. 35 U.S.C. §103 Rejection

Manning in view of Morgenstern

The Examiner has rejected claims 1-3, 7-13, 15-20 and 22-23 under 35 U.S.C. §103(a) as being unpatentable over Manning, U.S. Patent Publication No. 2002/0103829, ("Manning"), in view of Morgenstern, U.S. Patent No. 5,970,490 ("Morgenstern").

Manning shows an indexing system for aiding in queries. As stated in the abstract, one entry is added to at least one table. These tables, shown in Figure 6, are used in executing queries. The tables index the contents of XML documents (paragraph 6, line3, see also paragraph 9). and facilitate flexible queries to extract data from real and virtual documents (paragraph 7). There is no mention of persistence packages, persistent

data, nor transforms being applied to format data. Instead, Manning adds tables with pointers to the undisturbed XML data to be used for queries in an RDBMS.

In Claim 1, as amended, for example, persistent data is extracted from the persistence package. This data is then formatted and stored accordingly. In Manning, there is no formatting of the data before it is stored. Instead, some of the metadata is copied into tables. Accordingly, Claim 1 is believed to be allowable over the reference.

In paragraph 3.1, the Examiner sets forth an interpretation of Manning as showing a persistence package, persistent data, and a storage format. This interpretation does not come from the reference. However, due to the Examiner's persistence in this argument, Applicant will discuss the reference as if the claims applied to Manning.

In paragraph 3.2, the Examiner continues to draw the analogy in terminology between the claims and Manning as applied to "transforms". The Examiner further suggests that the values in the page table are formatted or transformed because the page table 220 values are numbers in the NUMBER column rather than text. Applicant submits that this is a table of page numbers. The values in the original document would be page numbers, just as they are page numbers in the NUMBER column. This may be compared to the text object table in which the data values are indicated as being text. The page table 220 is further illuminated by reference to Figure 5 in which there is an item 1006, identified as page number and the value is "2". "2" is the number from the document that will be added to the number field in the page table.

In paragraph 3.3, the Examiner continues that the NUMBER column shows that a value has been transformed or formatted as a number rather than as text. Again, Applicant submits that the value 1006 in the original is also a number not text.

By contrast, in Claim 1, the persistence package is in a format of a software component and is transformed into a storage format that is independent of the software component that provided the data. In Manning, the data is in XML, an application-independent language that is designed to be used by a wide range of different applications. It is not formatted for any particular software component. It is then converted into SQL tables, another application-independent structure.

In Claim 1, the data is provided from one of a plurality of different software components having persistent data in different formats. In Manning, the data is all in the same XML format no matter where it came from.

In Claim 1, the data is transformed without using the software component from which the persistence package is received. In Manning, different software components are not identified, but it is probable that the entire system is designed to work with the same XML formatted documents using the same software components.

Also in Claim 1, the storage format is a format that is compatible with the receiving system and with a storage device independent of the software component from which the persistent package was received. Manning does not say whether any software components are to be used but many SQL systems also handle XML.

As has been previously explained, Manning does not show formatting data before it is stored. Instead, in Manning some of the metadata is copied into tables. The Examiner's response to this fact of the reference is that "it would have been obvious." The Examiner's argument now asserted through more than 100 pages of written Office actions is to take words in Manning out of context and fill in with "it would have been obvious."

The Examiner's "it would have been obvious" approach is discussed in more detail below:

Claim 1 of the present application begins with receiving a persistence package from one of a plurality of different software components, the software components having persistent data in different formats. First, the Examiner has ignored the words "persistence" and "persistent" which have no parallel in Manning. Second, the Examiner has inferred the software components.

Most importantly, the Examiner asserts that ""it would have been obvious that persistent data from vector graphics is in a different format than e-commerce transactions." Applicant shall assume that the Examiner means to take Official Notice that data from vector graphics is in a different format than data from e-commerce transactions. Be that as it may, in Manning all the data is in XML + DTD format. There is nothing in the reference to suggest that there are different software components with different data formats.

Moving on to Claim 1's recitation of "establishing, based on the extracted metadata, a transform for a storage format for the persistent data," the Examiner leaves out the word "transform" and points to paragraphs 28 and 41. These paragraphs refer only to creating tables. The tables contain information taken from the XML data. The tables allow for querying and do not appear to have anything to do with persistence or different software applications, nor with transforms. The tables would appear to be additional metadata (para. 28, lines 3-4). At this point, Applicant is uncertain what "transform," and "storage format" are being read on. "Based on the extracted metadata" seems to have been ignored.

Claim 1 next recites, "applying the transform to the persistent data to format the persistent data." Manning does no such thing. The data is untouched, it is complemented by the element directory table and the navigation table, there is no transform to apply and no formatting.

Claim 1 further defines this formatting to be "from the format of the software component into a storage format." As mentioned previously, the Manning data comes in as XML and stays as XML with a little more metadata.

The Examiner has responded that "it would have been obvious that a transform is applied" Applicant believes that the Examiner means to assert that it is inherent in Manning that when a value is stored in an element table, a transform is applied to the value to determine where in the table to store that value. Of course, there is no suggestion in Manning that a transform be used. It would appear that the tables are added to the XML and that the database uses the XML tables.

As to the storing element of Claim 1, the Examiner would appear to be asserting that it is inherent in Manning that the tables are stored, that the storing is done in a storage device and that this is done during runtime. There are limitations regarding persistence, software components, and the storage device that all interrelate and that are being ignored.

For further support regarding the transforms, the Examiner has turned to Morgenstern at Col. 6, line 1, through Col. 8, line 53. Applicant is unable to find a transform for a storage format for persistent data in this section of Morgenstern. Applicant is further unable to find, and the Examiner is unable to point out, any suggestion that transformation generator 42 establishes a transform based on metadata or operates without using the software components from which a data package is received.

On the contrary, it would appear that the source is tightly connected with the transform generator and transformer engine 66.

Further, limitations in the claims additionally distinguish the invention from the references. Here are a few:

"foreign to the running system

"transforms establish a storage format and/or "storage location"

"description of the format of the persistent data"

Applicant respectfully submits that these additional features reflect features not present in the cited combination and accordingly, the rejection is respectfully traversed. For example, the XML documents require a specific program to parse the XML. This program is normally also able to edit, display, print, and convert the documents and style sheets, etc. The software is thus not foreign to the system.

Manning and Morgenstern are both directed to indexing systems for queries in a relational database. The invention of Claim 1 is directed, at least in part, to formatting data from the format of the software component from which it was received into a storage format that is compatible with the receiving system. This is done without using the software component.

Accordingly, the elements of Claim 1 are not met by the cited combination. Claims 29 and 36 were rejected on similar grounds. These claims contain even more limitations that are neither taught nor suggested by the combination.

VIII. CONCLUSION

Appellant respectfully submits that all appealed claims in this application are patentable and were improperly rejected by the Examiner during prosecution before the United States Patent and Trademark Office. Appellant respectfully requests that the Board of Patent Appeals and Interferences overrule the Examiner and direct allowance of the rejected claims.

This Brief is submitted with a check for $500.00 to cover the appeal fee for one other than a small entity as specified in 37 C.F.R. § 1.17(c). Please charge any shortages and credit any overpayments to our Deposit Account No. 02-2666.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Dated: December 4, 2007

Gordon R. Lindeen III
Reg. No. 33,192

12400 Wilshire Boulevard
Seventh Floor
Los Angeles, CA. 90025-1030
(303) 740-1980

IX. APPENDIX OF CLAIMS (37 C.F.R. § 41.37(c)(1)(viii))

1. A method, comprising:

receiving a persistence package at a running system from one of a plurality of different software components, the persistence package including persistent data and metadata, the software components having persistent data in different formats that are foreign to the running system;

extracting persistent data and metadata from the persistence package, the persistent data relating to diverse types of objects constructed at runtime of the software component and needed during more than one invocation of the software component, the metadata describing the persistent data, and comprising, at least in part, a description of the format of the persistent data ;

establishing, based on the extracted metadata, a transform for a storage format and a storage location for the persistent data ;

applying the transform to the persistent data to format the persistent data without using the software component from which the persistence package was received from the format of the software component into a storage format that is compatible with the receiving system and with a storage device of the running system independent of the software component; and

storing the persistent data in the storage device in the storage format .

2. The method of claim 1, further comprising using metadata passed from the persistence package to establish a storage location for the persistent data during the runtime of the system.

3. The method of claim 1, wherein the metadata comprises at least in part a description of a model structure of the persistent data.

4. The method of claim 3, wherein the metadata conforms to a metadata template comprising rules for describing the model structure.

5. The method of claim 4, wherein extracting the persistent data and the metadata from the persistence package comprises using a filter.

6. (Canceled)

7. The method of claim 1, further comprising retrieving persistent data from storage using a transform during the runtime of the receiving system.

8. The method of claim 1, further comprising receiving persistent data compatible with at least one of any type of processor, any type of programming language, any type of operating system, and any type of architecture.

9. An apparatus, comprising:

a data storage device;

a running receiving system coupled to the data storage device, the receiving system including a persistence engine to receive a persistence package from one of a plurality of different software components that are foreign to the running system, the persistence package including persistent data and metadata, the metadata comprising, at least in part, a description of the format of the persistent data the software components having persistent data in different formats, wherein the persistence engine extracts persistent data and metadata from the persistence package, wherein the persistence engine uses the extracted metadata passed from the persistence package to establish, without using the software component from which the persistence package was received, a storage format and location to store the persistent data in the data storage device, and wherein the persistence engine applies the storage format to the persistent data to format the persistent data from the format of the software component into a storage format that is compatible with the receiving system and with the storage device independent of the software component, and to store the formatted system data in the data storage device.

10. The apparatus of claim 9, wherein the data storage device is external to a receiving system using the persistence engine.

11. The apparatus of claim 9, further comprising a storing interface to store the persistent data using the storage format.

12. The apparatus of claim 9, further comprising a retrieving interface to retrieve stored persistent data for use by one of the receiving system and the software component, the software component comprising an application.

13. The apparatus of claim 9, wherein the metadata comprises at least in part a description of the data model structure of the persistent data.

14. The apparatus of claim 13, further comprising a metadata template to format the metadata for readable reception by the persistence engine.

15. The apparatus of claim 9, wherein the persistence engine receives a persistence package comprising the metadata and the persistent data.

16. The apparatus of claim 9, wherein the persistence engine receives persistent data structured using any data model from a source comprising at least one of any type of processor, any type of operating system, any type of programming language, and any type of architecture.

17. The apparatus of claim 9, further comprising a metadata engine having a metadata reader and a metadata filter.

18. The apparatus of claim 17, wherein the metadata filter interprets the metadata.

19. The apparatus of claim 9, further comprising a transform engine having a set of transforms, a transform selector, and a transform generator.

20. The apparatus of claim 19, wherein a transform establishes at least one of the storage format and the storage location to store the persistent data in the data storage device.

21. The apparatus of claim 19, the transform selector further comprising a data model comparator.

22. The apparatus of claim 19, wherein the transform selector selects a transform based on filtered metadata.

23. The apparatus of claim 19, wherein the transform selector requests a transform from the transform generator based on filtered metadata.

24. The apparatus of claim 23, wherein the transform generator produces a transform that remodels the persistent data to approximate as closely as possible a preexisting transform from the set of transforms.

25. The apparatus of claim 23, wherein the transform generator produces a transform that substantially maintains the model structure of the persistent data received by the receiving system.

26. The apparatus of claim 23, wherein the transform generator produces a transform to remodel the persistent data to maximize efficient retrieval for an application.

27. The apparatus of claim 23, wherein the transform generator uses iterative read-write trials to produce a transform to remodel the persistent data to maximize storage and/or retrieval speed.

28. The apparatus of claim 23, wherein the transform generator produces a transform to remodel the persistent data to maximize data compression.

29. An apparatus, comprising:

a communications interface;

a data model description receiver to receive a data model description from one of a plurality of different software components that are foreign to the apparatus, the software components having persistent data in accordance with different data models, the data model descriptions describing, at least in part a format of data associated with the models;

a set of transforms;

a data model comparator to produce a comparison independent of the software component from which the data model description is received between the data model description and a data model in a transform in the set of transforms;

a transform generator having an assembler to produce a transform based on the data model description and the comparison independent of the software component from which the data model description was received, the transform establishing a storage format and a storage location for data associated with the model;

a storage device;

a transform engine to apply a transform to format persistent data for storage from the format of the software component into a storage format that is compatible with the storage device independent of the software component; and

a storing interface to store the formatted persistent data in the storage device.

30. The apparatus of claim 29, wherein the transform generator further comprises a data model variance calculator coupled to the assembler.

31. The apparatus of claim 29, wherein the transform generator further comprises a data model approximator coupled to the assembler.

32. The apparatus of claim 29, wherein the transform generator further comprises an efficient storage/retrieval speed maximizer coupled to the assembler.

33. The apparatus of claim 32, wherein the storage/retrieval speed maximizer further comprises a read/write iterator.

34. The apparatus of claim 29, wherein the transform generator further comprises a data compression maximizer coupled to the assembler.

35. The apparatus of claim 29, wherein the transform generator further comprises an indexing estimator coupled to the assembler.

36. A method, comprising:

receiving a data model description at a running system from one of a plurality of different software components that are foreign to the running system, the software components having persistent data in accordance with different data models, the persistent data relating to diverse types of objects, the data model description describing the persistent data, and comprising, at least in part, a description of the format of the persistent data;

comparing the data model description to a preexisting data model independent of the software component from which the data model description is received;

assembling a transform at the running system independent of the software component from which the data model description is received based on the data model description and the comparison to establish a storage format for persistent data during runtime of a system;

applying a transform to format persistent data for storage from the format of the software component into a storage format that is compatible with a storage device independent of the software component; and

storing the formatted persistent data at the storage device.

37. The method of claim 36, wherein the assembling a transform includes measuring a variance between the data model description and a preexisting data model.

38. The method of claim 36, wherein the assembling a transform includes approximating a preexisting data model.

39. The method of claim 36, wherein the assembling a transform includes maximizing data storage speed and/or data retrieval speed.

40. The method of claim 39, wherein the maximizing speed includes iteratively performing data read/write trials and selecting the fastest trial.

41. The method of claim 36, wherein the assembling a transform includes maximizing data compression.

42. The method of claim 36, wherein the assembling a transform includes optimizing efficient indexing for the persistent data.

43. An article of manufacture, comprising:

a machine-readable medium comprising instructions, that when executed cause a machine to:

receive persistent data having a model structure from one of a plurality of different software components that are foreign to the machine and the machine-readable medium, the software components having persistent data in different model structures, the persistent data relating to diverse types of objects ;

receive metadata comprising at least in part a description of the model structure, the metadata describing the persistent data and comprising, at least in part, a description of the format of the persistent data ; and

establish, using the metadata and without using the software component from which the persistence package was received, a storage format and a storage location for the persistent data; and

apply the established storage format to the persistent data to format the persistent data for storage from the format of the software component into a storage format that is

compatible with the machine and with a storage device independent of the software component.

44. The article of manufacture of claim 43, further comprising instructions, that when executed, cause a machine to store the persistent data using the storage format.

45. The article of manufacture of claim 43, further comprising instructions, that when executed, cause a machine to receive metadata conforming to a metadata template comprising rules for describing a data model structure of the persistent data.

46. The article of manufacture of claim 45, further comprising instructions, that when executed, cause a machine to receive a persistence package comprising the persistent data and the metadata and to extract the persistent data and the metadata from the persistence package.

47. The article of manufacture of claim 43, further comprising instructions, that when executed, cause a machine to retrieve the persistent data using the storage format.

48. The article of manufacture of claim 43, further comprising instructions, that when executed, cause a machine to select and/or create, based on the metadata, a transform to establish at least one of the storage format and the storage location.

49. The article of manufacture of claim 43, further comprising instructions, that when executed, cause a machine to receive persistent data compatible with one of any type of processor, any type of programming language, any type of operating system, and any type of architecture.

X. EVIDENCE APPENDIX

None.

XI. RELATED PROCEEDINGS APPENDIX

None.